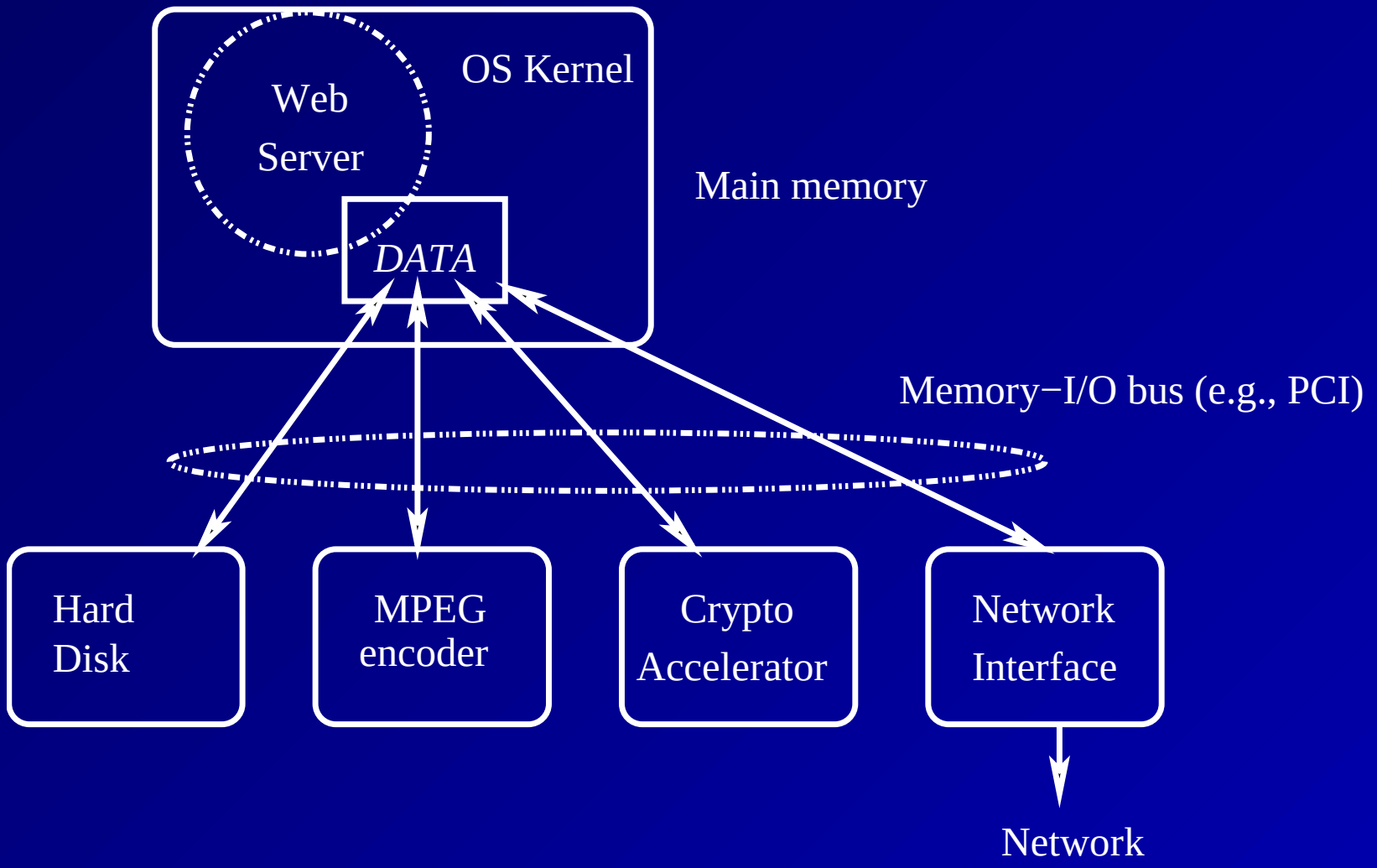
Adding a Flow-Oriented Paradigm to Commodity Operating Systems

Cristian Soviani, Stephen A. Edwards, and Angelos Keromytis

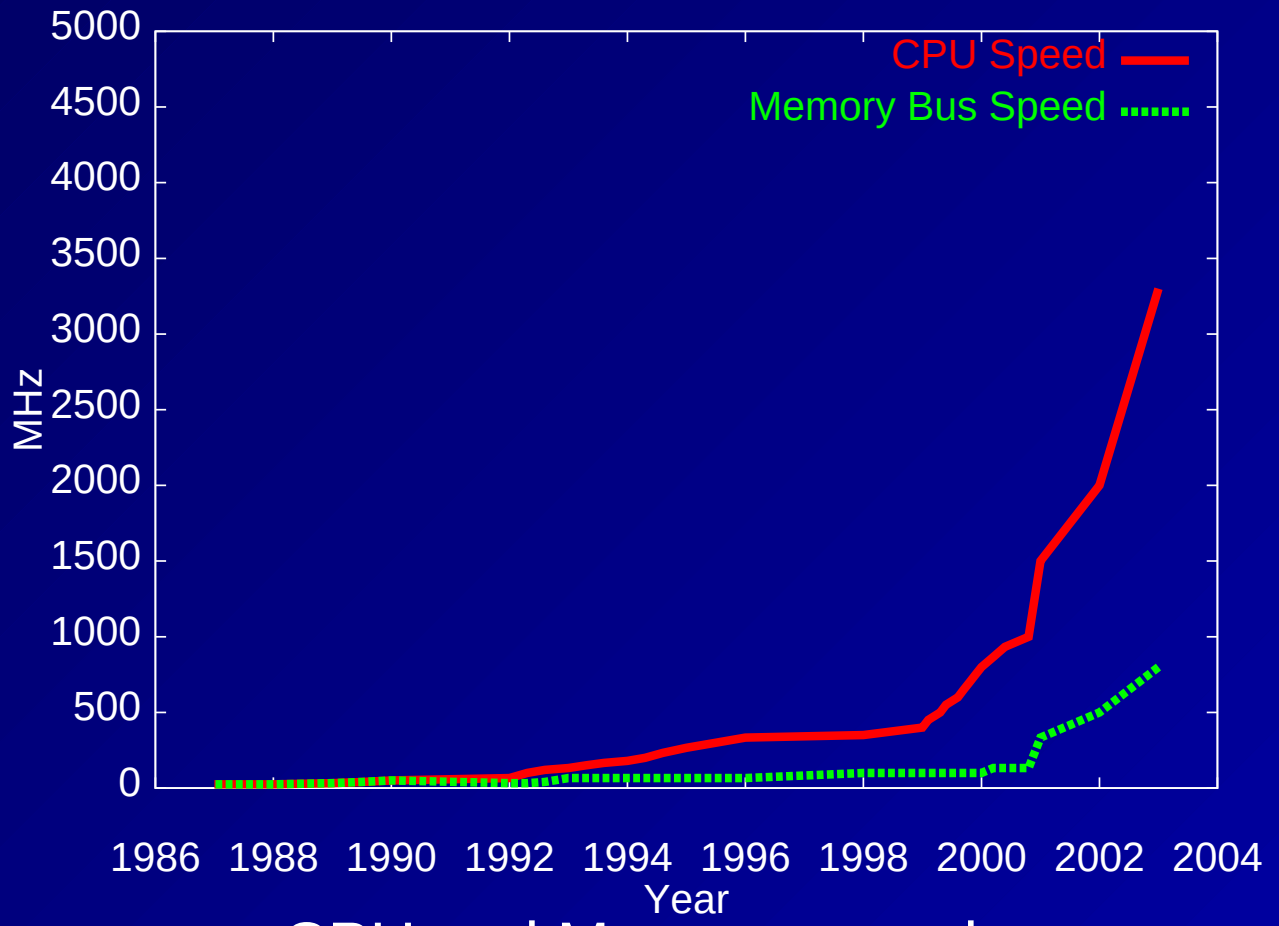
Department of Computer Science,
Columbia University in the City of New York

{soviani,sedwards,angelos}@cs.columbia.edu

The Status Quo: Memory as Buffer



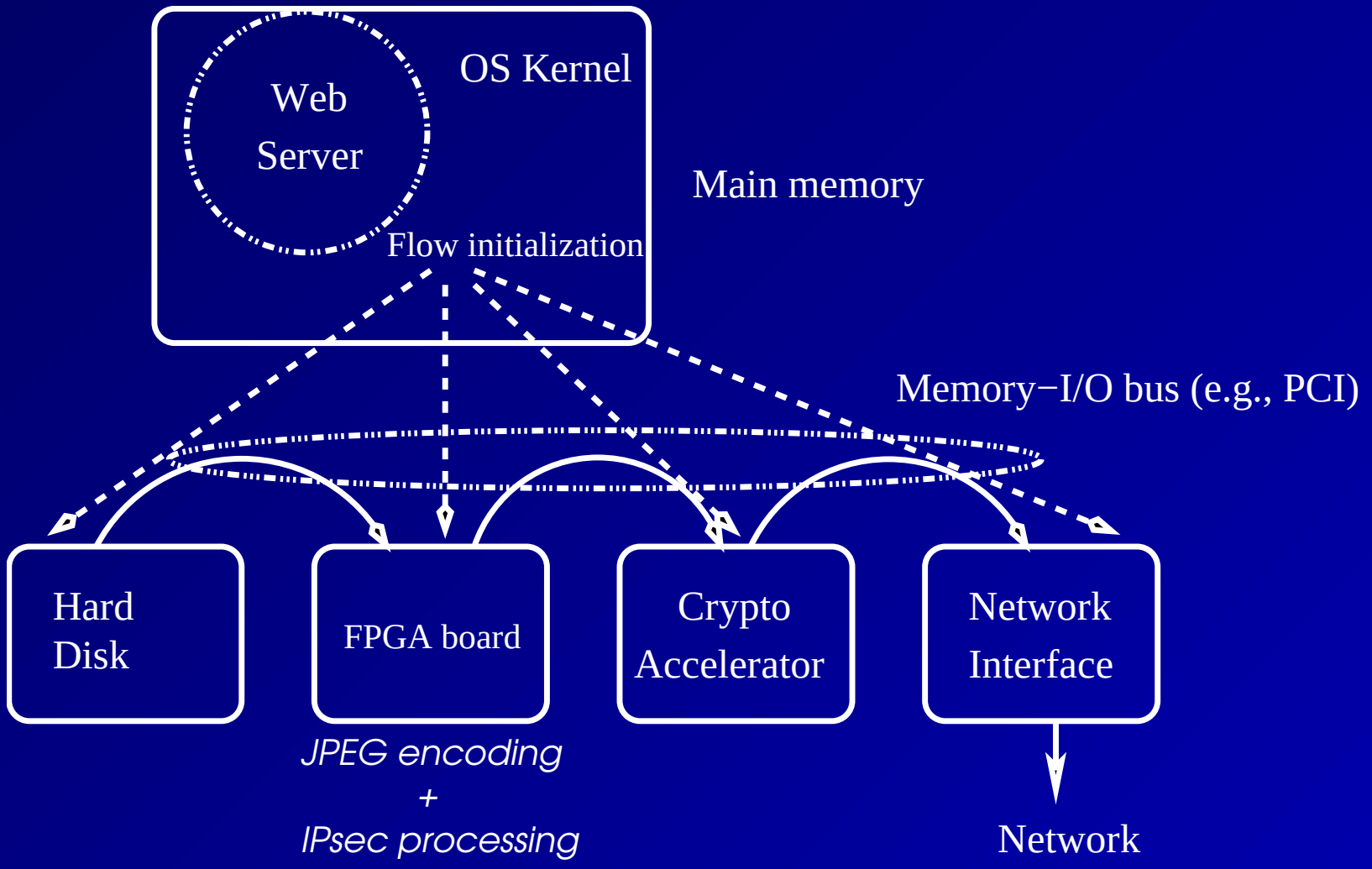
I/O Becoming Faster than CPUs



CPU and Memory speeds

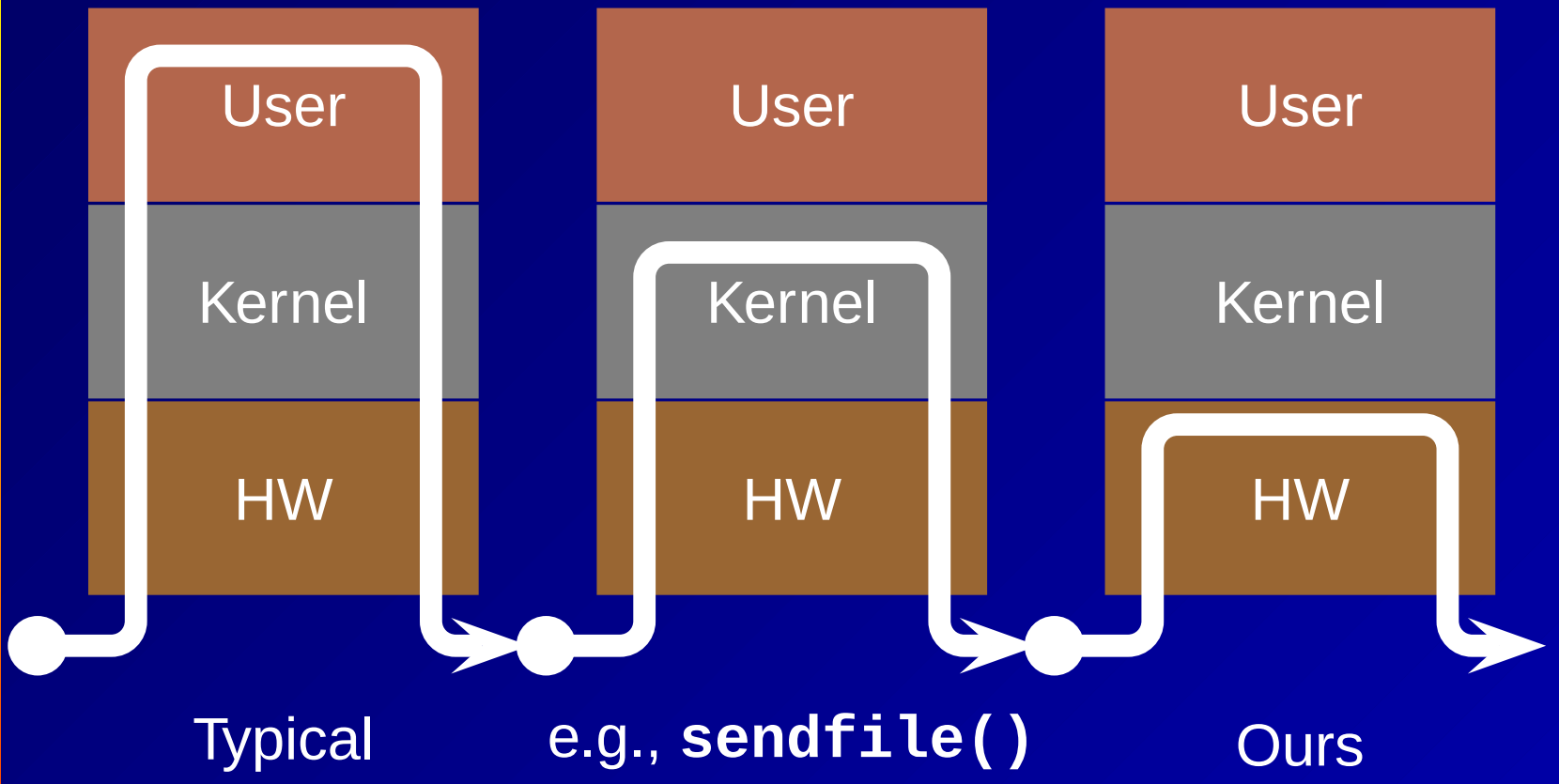
(Source: Intel)

Our Idea: Flows Controlled by OS

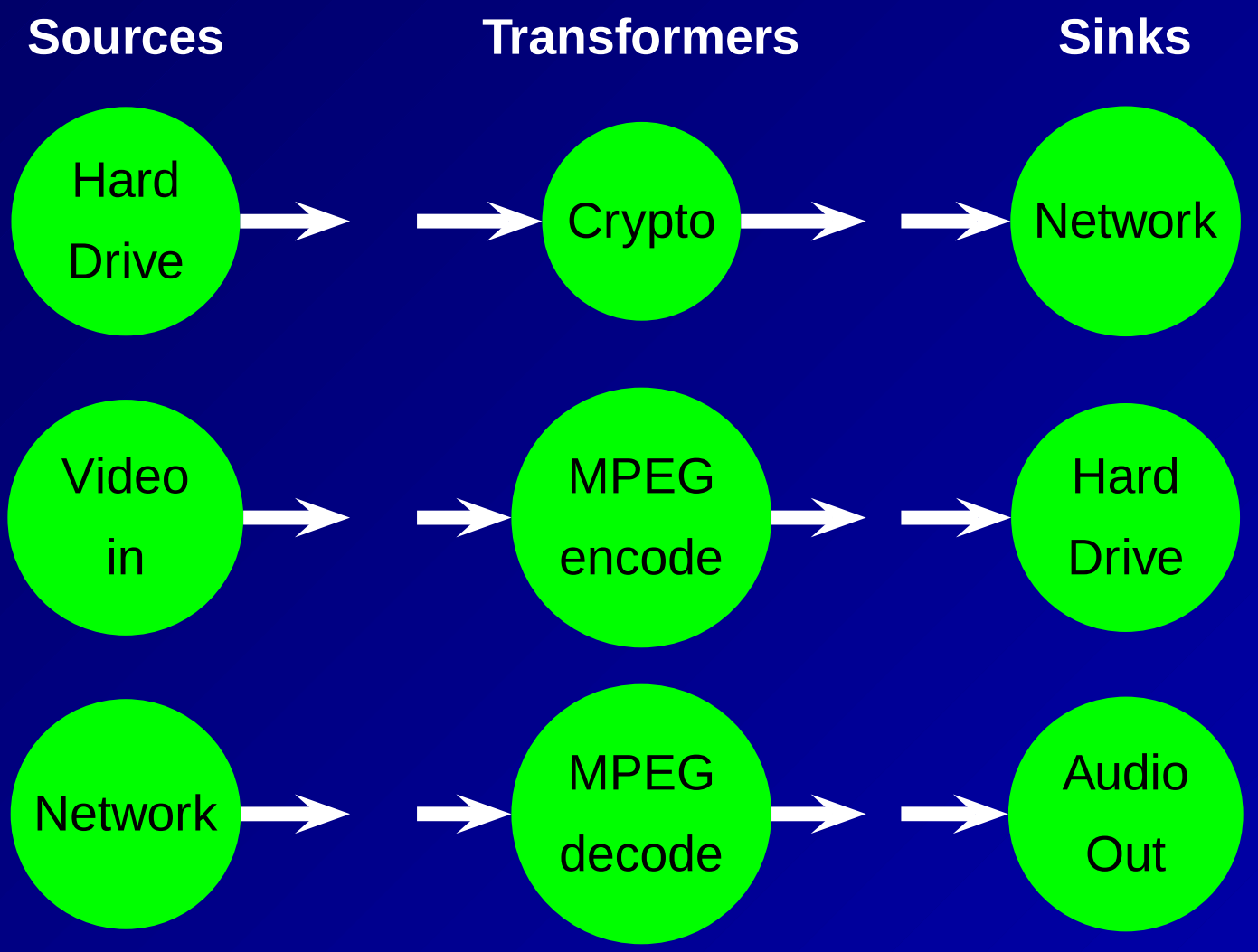


Like a modern router's control & data plane

Layers


User
Kernel
HW
User
Kernel
HW
User
Kernel
HW
Typical
e.g., sendfile()
Ours

Abstract Server Model



Signaling API

Need some sort of mechanism for creating, controlling, and tearing down flows, e.g.,

```
flow = flow_open();
flow_source(flow, "/usr/http/secretfile.html");
flow_xformer(flow, "/dev/crypto");
flow_xformer(flow, "/dev/http");
flow_sink(flow, "/dev/inet/192.168.1.3");
flow_start(flow);
flow_stop(flow);
```

Exception Handling

What happens if something goes wrong?

- OS may try to re-start device
- Redirect to a different device
- Error passed to application
- Switch to all-software flow
- Terminate

Resource Scheduler

What if two processes want access to /dev/crypto?

- Performance requirements
- Available resources
- Priorities

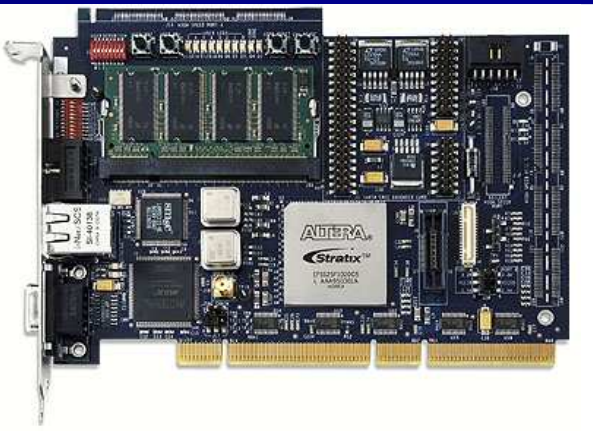
Flow-level scheduling costly, but infrequent

Detailed (e.g., bus access) scheduling more frequent

Programmable Peripherals

RadiSys ENP-2611

Network Card w/ IXP2400

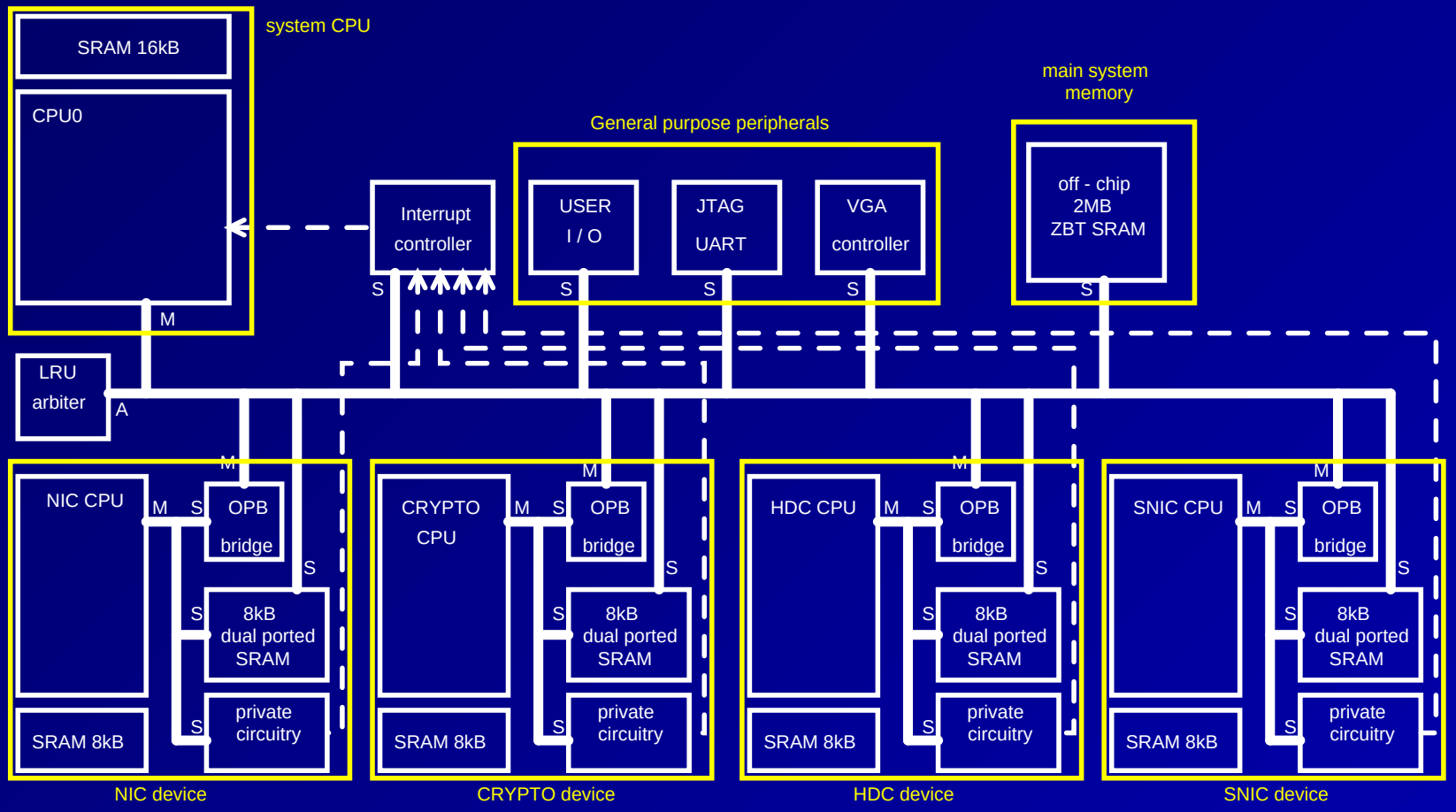


Altera Stratix PCI

FPGAs: flow components or

wrappers around legacy peripherals (DMA absorbers)

Proof-of-Concept System



Experimental Results

Time to send 1 Million packets through the pipeline

Packet size	Memory-centric	Flow-centric*	Speed-up
64 bytes	15.5s	13.6	13%
1024 bytes	114	59.2	49%

*Plus main processor mostly idle for these packets

Conclusions

New flow-centric architecture for operating systems

Have the OS manage inter-peripheral flows under application control

Requires programmable peripherals: many already extant

Proof-of-concept showed nearly a $2\times$ speedup

Minimal performance impact from additional computations (e.g., security)